//+------------------------------------------------------------------+

//| YTS.mq4 |

//| Copyright 2023, YourName |

//| https://www.yourwebsite.com |

//+------------------------------------------------------------------+

// Global variables

int EMA\_Period = 14;

int SMA\_Period = 50;

int ADX\_Period = 14;

int RSI\_Period = 14;

double Stoch\_Period = 14;

double Stoch\_Smooth\_K = 3;

double Stoch\_Smooth\_D = 3;

int BB\_Period = 20;

double BB\_Deviation = 2.0;

int Momentum\_Period = 14;

int MACD\_Fast\_EMA = 12;

int MACD\_Slow\_EMA = 26;

int MACD\_Signal\_SMA = 9;

double ADX\_Level\_Buy = 25;

double ADX\_Level\_Sell = 25;

double RSI\_Level\_Buy = 70;

double RSI\_Level\_Sell = 30;

double Stoch\_Oversold = 20;

double Stoch\_Overbought = 80;

// Arrays

double ADR\_Buffer[];

double EMA\_Buffer[];

double SMA\_Buffer[];

double ADX\_Buffer[];

double RSI\_Buffer[];

double Stoch\_K[];

double Stoch\_D[];

double BB\_Upper[];

double BB\_Lower[];

double Momentum\_Buffer[];

double MACD\_Buffer[];

double MACD\_Signal\_Buffer[];

bool isBuySignal = false;

bool isSellSignal = false;

bool isTradeExecuted = false;

double TP1\_Pips = 10;

double TP2\_Pips = 20;

double SL\_Pips = 10;

// Trade colors

color BuyColor = clrBlue;

color SellColor = clrRed;

// +------------------------------------------------------------------+

// | Expert initialization function |

// +------------------------------------------------------------------+

int OnInit()

{

// Initialization procedures go here

// Set parameters based on hourly, 4-hour, and daily timeframes

SetParametersByTimeframe();

// Initialize ADR buffer

ArrayResize(ADR\_Buffer, Bars);

return(INIT\_SUCCEEDED);

}

// +------------------------------------------------------------------+

// | Expert tick function |

// +------------------------------------------------------------------+

void OnTick()

{

// Tick operations go here

// Check trading conditions and execute trades

CalculateIndicators();

ExecuteTrade();

}

// +------------------------------------------------------------------+

// | Function to Set Parameters Based on Timeframe |

// +------------------------------------------------------------------+

void SetParametersByTimeframe()

{

if (Period() == PERIOD\_H1)

{

EMA\_Period = 9;

SMA\_Period = 20;

ADX\_Period = 14;

RSI\_Period = 14;

Stoch\_Period = 14;

Stoch\_Smooth\_K = 3;

Stoch\_Smooth\_D = 3;

Stoch\_Oversold = 20;

Stoch\_Overbought = 80;

BB\_Period = 20;

BB\_Deviation = 2;

Momentum\_Period = 14;

MACD\_Fast\_EMA = 12;

MACD\_Slow\_EMA = 26;

MACD\_Signal\_SMA = 9;

}

else if (Period() == PERIOD\_H4)

{

EMA\_Period = 40;

SMA\_Period = 100;

ADX\_Period = 14;

RSI\_Period = 14;

Stoch\_Period = 14;

Stoch\_Smooth\_K = 3;

Stoch\_Smooth\_D = 3;

Stoch\_Oversold = 20;

Stoch\_Overbought = 80;

BB\_Period = 20;

BB\_Deviation = 2;

Momentum\_Period = 14;

MACD\_Fast\_EMA = 12;

MACD\_Slow\_EMA = 26;

MACD\_Signal\_SMA = 9;

}

else if (Period() == PERIOD\_D1)

{

EMA\_Period = 80;

SMA\_Period = 150;

ADX\_Period = 14;

RSI\_Period = 14;

Stoch\_Period = 14;

Stoch\_Smooth\_K = 3;

Stoch\_Smooth\_D = 3;

Stoch\_Oversold = 20;

Stoch\_Overbought = 80;

BB\_Period = 20;

BB\_Deviation = 2;

Momentum\_Period = 14;

MACD\_Fast\_EMA = 12;

MACD\_Slow\_EMA = 26;

MACD\_Signal\_SMA = 9;

}

}

// +------------------------------------------------------------------+

// | Function to Execute Trade |

// +------------------------------------------------------------------+

void ExecuteTrade()

{

// Check trade conditions and execute trades

if (CheckTradeConditions())

{

// Execute trade logic

double lotSize = CalculateLotSize();

double tp1 = TP1\_Pips \* \_Point;

double tp2 = TP2\_Pips \* \_Point;

double sl = SL\_Pips \* \_Point;

if (isBuySignal)

PlaceBuyOrder(lotSize, tp1, tp2, sl);

else if (isSellSignal)

PlaceSellOrder(lotSize, tp1, tp2, sl);

}

}

// +------------------------------------------------------------------+

// | Function to Set Trade Execution Status |

// +------------------------------------------------------------------+

void SetTradeExecuted(bool executed)

{

isTradeExecuted = executed;

}

// +------------------------------------------------------------------+

// | Function to Calculate Lot Size |

// +------------------------------------------------------------------+

double CalculateLotSize()

{

// Risk percentage for each trade

double riskPercentage = 2.0;

// Calculate lot size based on risk percentage

double lotSize = AccountFreeMarginCheck(\_Symbol, OP\_BUY, 0.1) \* riskPercentage / 100 / SL\_Pips / \_Point;

// Normalize lot size to 2 decimal places

return NormalizeDouble(lotSize, 2);

}

// +------------------------------------------------------------------+

// | Function to Calculate Average Daily Range (ADR) |

// +------------------------------------------------------------------+

double CalculateADR(int period)

{

// Implement ADR calculation logic

double adr = iATR(\_Symbol, \_Period, period, 0);

return adr;

}

// +------------------------------------------------------------------+

// | Function to Calculate Indicators |

// +------------------------------------------------------------------+

void CalculateIndicators()

{

// Implement indicator calculation logic here

// ...

}

// +------------------------------------------------------------------+

// | Function to Check Buy Signal |

// +------------------------------------------------------------------+

bool IsBuySignal()

{

// Implement Buy signal logic here

// ...

return false;

}

// +------------------------------------------------------------------+

// | Function to Check Sell Signal |

// +------------------------------------------------------------------+

bool IsSellSignal()

{

// Implement Sell signal logic here

// ...

return false;

}

// +------------------------------------------------------------------+

// | Function to Check if Trade is Executed |

// +------------------------------------------------------------------+

bool IsTradeExecuted()

{

// Implement trade execution check logic here

// ...

return false;

}

// +------------------------------------------------------------------+

// | Function to Place Buy Order |

// +------------------------------------------------------------------+

void PlaceBuyOrder(double lotSize, double tp1, double tp2, double sl)

{

// Buy order logic

int ticket = OrderSend(\_Symbol, OP\_BUY, lotSize, Ask, 3, 0, 0, "Buy Order", 0, 0, BuyColor);

// If the trade is executed successfully

if (ticket > 0)

{

// Place Take Profit and Stop Loss orders

PlaceTakeProfitAndStopLoss(ticket, tp1, tp2, sl, BuyColor);

}

else

{

Print("Buy order failed! Error code: ", GetLastError());

// You can perform additional operations here if needed.

}

}

// +------------------------------------------------------------------+

// | Function to Place Sell Order |

// +------------------------------------------------------------------+

void PlaceSellOrder(double lotSize, double tp1, double tp2, double sl)

{

// Sell order logic

int ticket = OrderSend(\_Symbol, OP\_SELL, lotSize, Bid, 3, 0, 0, "Sell Order", 0, 0, SellColor);

// If the trade is executed successfully

if (ticket > 0)

{

// Place Take Profit and Stop Loss orders

PlaceTakeProfitAndStopLoss(ticket, tp1, tp2, sl, SellColor);

}

else

{

Print("Sell order failed! Error code: ", GetLastError());

// You can perform additional operations here if needed.

}

}

// +------------------------------------------------------------------+

// | Function to Place Take Profit and Stop Loss Orders |

// +------------------------------------------------------------------+

void PlaceTakeProfitAndStopLoss(int ticket, double tp1, double tp2, double sl, color color)

{

// Take Profit 1

double takeProfit1Price = Ask + tp1;

int takeProfit1Ticket = OrderSend(\_Symbol, OP\_TAKEPROFIT, ticket, takeProfit1Price, 3, 0, 0, "Take Profit 1", 0, 0, color);

// Take Profit 2

double takeProfit2Price = Ask + tp2;

int takeProfit2Ticket = OrderSend(\_Symbol, OP\_TAKEPROFIT, ticket, takeProfit2Price, 3, 0, 0, "Take Profit 2", 0, 0, color);

// Stop Loss

double stopLossPrice = Bid - sl;

int stopLossTicket = OrderSend(\_Symbol, OP\_STOPLOSS, ticket, stopLossPrice, 3, 0, 0, "Stop Loss", 0, 0, color);

}